CHAPTER 1

Welcome to React

React is a popular library for creating user interfaces. It was created at Facebook to address some of the challenges associated with large-scale, data-driven websites. When React was released in 2013, the project was initially viewed with some skepticism because the conventions of React are quite unique.

In an attempt to not scare people off, the core React team wrote an article called Why React that recommended that you “Give It [React] Five Minutes”. Their point was that they wanted to encourage people to work with React first before thinking that their approach was too crazy.

Yes, React is a small library that doesn’t come with everything you might need out of the box to build your application. Give it five minutes.

Yes, in React, you write code that looks like HTML right in your JavaScript code. And yes, those tags require pre-processing to run in a browser. And, you’ll probably need a build tool like webpack or browserify for that. Give it five minutes.

Reading this book won’t take you five minutes, but we do invite you to dive into React with an open mind.

A few companies that have given React more than five minutes, and use the library for large parts of their web interfaces, include Airbnb, Khan Academy, and the New York Times[1]. Many of Facebook’s features and all of Instagram[2] are built on React and

1 These companies were early adopters of React and used it in production as early as 2014. https://facebook.github.io/react/blog/2014/05/29/one-year-of-open-source-react.html

2 “Why Did We Build React” by Pete Hunt, Facebook Blog: https://facebook.github.io/react/blog/2013/06/05/why-react.html

associated tools to manage the messages and pictures of lunch that over a billion users[3]post every day.

The widespread use of React on large websites shows that it is stable enough to use at scale. React is ready, but nothing is set in stone. The unique opportunity we all have is that since it's so new, we can be part of building it. As the library and its tools evolve, we can suggest enhancements. When ideas come to mind about how to work with React more efficiently, we can build them. React is already great, but we can play an active role in building its even better future.

History

React was built at Facebook and Instagram, released initially in March 2013, then open-sourced on May 29, 2013. React is for your user interfaces or the view layer of your application. It was designed as a view-only library where you create user interface components that display data.

When it was released, React built steam quickly, and the community quickly contributed code and got involved in community events.

With all of the growth, Facebook decided to build and release a framework for building native applications in 2015[4] : React Native. You can build apps for iOS, Android, and Windows platforms using React Native. Unlike other platforms, React uses the native phone and tablet UI elements. The aim of React Native is to use the same programming language to build many types of apps - not necessarily the same codebase.

A design architecture that emerged around the same time as React from the Facebook team is Flux. Flux was built to deal with a problem in the Facebook messaging app. Users complained that when they read a message, they would still see a notification that they had an unread message[5]. To deal with these data inconsistencies, Flux introduced a new design where data flowed one way. This data flow works particularly well with React.

Building upon Flux's ideas, Redux was developed to simplify the process of managing data in React apps. It was released in 2015 and has picked up a lot of momentum as a less complex but similarly solid implementation of Flux. In addition, Falcor and Relay have also emerged to tackle data handling challenges.

3 There are over 1.65 billion monthly active Facebook users as of 04/27/16. https://zephoria.com/top-15-valuable-facebook-statistics/

4 Tom Occhino introduces React Native in 2015 at the React Conference. https://code.facebook.com/videos/786462671439502/react-js-conf-2015-keynote-introducing-react-native-/

5 Facebook engineer Jing Chen discusses this in her 2014 talk, "Rethinking Web App Development at Facebook". https://facebook.github.io/flux/docs/overview.html#content

At the time we're writing this book, React isn't the most popular JavaScript library, but it's growing fast. Web searches for JavaScript frameworks and libraries indicate a huge rise in popularity.

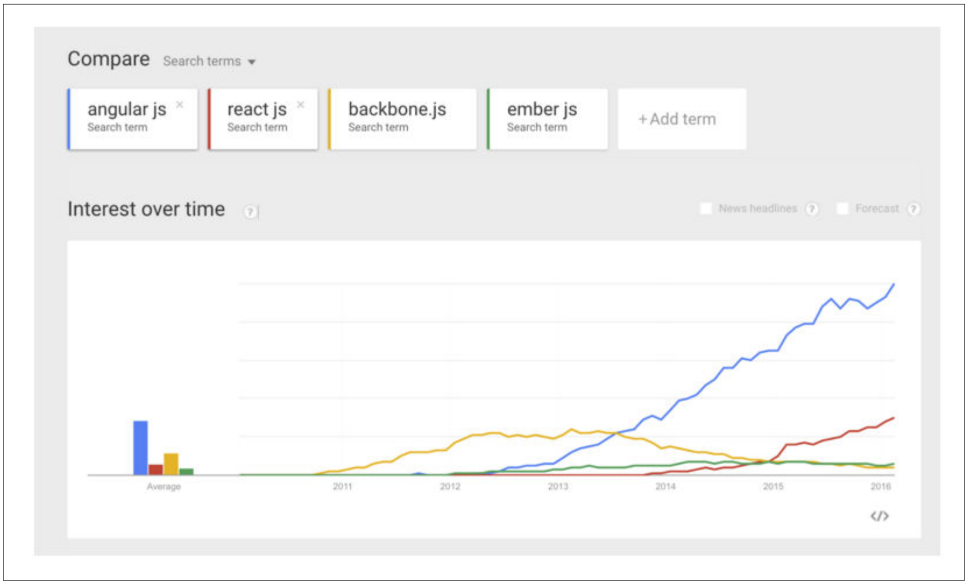


Figure 1-1.
Web Searches for JavaScript Frameworks & Libraries (Google Trends)

You've probably heard React described as the hot new library that everyone in the organization should learn to replace whatever everyone learned last month. Being popular isn't the reason to use React. Popularity is a side effect of being useful and time saving.

The influence of React is even felt in other MVC frameworks. Angular and Ember have been inspired by React's approach in newer versions of those frameworks. It's interesting to observe where there are overlaps and how the individual communities approach this work.

So, what does React do next? Perhaps we'll use React to build desktop apps. We might build console apps. We might use React to build robots or to make up the screens of our self-driving cars.

What will happen? None of us know, but you can join us. And, we need you to join us!

React is not a Framework

React is a library not a framework. A lot of people call it a framework or compare it to frameworks. We even compared it to frameworks in the previous section.

The reason is React is commonly mistaken as a framework is because it feels like frameworks are React's competitors. Most blog articles set up a classic bout between React vs. Angular or React vs. Ember. You can use React with Angular. You can use React with Ember. Although, for the most part, we typically do not combine Angular with React, you could.

Frameworks are bigger than libraries. They include everything that you may need to build applications. Think about the .NET framework. Not only Not only does it have everything you need to build any type of Microsoft application, but there are specific design patterns in .NET that you would follow to construct applications. The .NET framework even comes with its own IDE[6], Visual Studio.

Angular is a JavaScript framework that you can use to build large scale single page web applications. Angular comes with most everything you need to get some web development done. There are even design patterns to follow, primarily patterns based in MVC[7]. The Angular 2 JavaScript file that loads in the browser is 764k, but it is rich with functionality.

By comparison, the JavaScript file that is used to load React in the browser is 151k. React is small because it is simply a view library. It has some robust tools for managing views. It even has some tools to build full applications, but React relies a lot on pure JavaScript and other JavaScript libraries.

React does not come with any REST tools for making HTTP requests. React does not come with any tools to handle client-side routing. It is simply a library that helps you build efficient user interfaces.

React does not impose any restrictions on your data architecture. You can use React with MVC. You can use React with jQuery. You can use React with Flux or Redux. You can even build your own client data management library or design pattern and use React with that.

The reason that it is important to understand that React is only a library is because calling it a framework generates confusion about how to learn React. If you set out to learn Angular, the framework has everything you need to learn already included. It

6 IDE is an acronym for Integrated Development Environment and is a tool that you use to write code.

7 MVC is an object-oriented design pattern that can be used in any object-oriented language. MVC stands for Model View Controller. The model is the data, the view is the presentation layer, and the controller is the business logic.

also includes common design patterns that address how to build a web application. You can say, “I’m learning Angular 2”, and get started learning a single framework.

React is a little bit different. You could say, “I’m learning React”, learn how to build a React component, and then not know what to do next. The reason for this is because with React, you have some decisions that you need to make about your overall architecture. Is your app small enough to build with React and React alone? Is your application going to be a single page application, SPA? If so, what are you going to use for routing? If you’re using Flux, which flavor of Flux: Facebook Flux, Reflux, Redux, or your own implementation? How are you going to build or transpile your source code? Are you using emerging JavaScript like ES6 or ES7? You are going to need some answers to these questions before you get started learning, and that alone may feel daunting.

The good news is, if you answer these questions, working with React can be quite rewarding. To make learning React more approachable we are simply going to answer these questions for you and create a learning plan based on our answers. We are going to learn to use React alone and React with other libraries. We are not using MVC with React. We will build SPA’s, and we’ll even build universal applications[8] . We will introduce functional programming and follow functional principles throughout the book. We will introduce the Flux pattern, and we’ll build applications using Redux, an implementation of the Flux pattern. We will use webpack and Babel to transpile our code. We are heavily using emerging JavaScript like ES6 and ES7.

React and MVC

In the previous section, we mentioned that React can be used with MVC. Originally, when React was first introduced, some developers even referred to it as the “V” in MVC. When we were first introduced to React, the first several applications that we developed used Backbone models, collections, and routes. If you are very familiar with MVC, you can probably hit the ground running and build some robust applications that use React with your favorite MVC framework.

One approach to learning React is to build some small components and swap them out with existing views in your project. React does not demand that you change your entire infrastructure. You can try React components out in your present MVC application without too much work.

React was developed to handle problems that can arise out of building MVC applications. If you use React with MVC, sooner or later you may encounter these problems. They start to show up when we have models that interact with multiple views, and

8 Universal apps are applications that use the same code on both the client and the server. See Chapter 12 for more.

views that listen to models to update UI when models change. In short MVC causes side effects. A certain view that you didn't expect to change could have changed when a certain model was updated. Sooner or later you may find complications within your React app that will push you to learn Flux. That is what happened to us

Learning Flux is a journey into functional JavaScript is going to deepen your knowledge of the JavaScript language. React is going to provide a user interface as data abstraction. Most of your work will be with the core datasets. These datasets will consist of JavaScript arrays and object literals. Functional programming demands that we keep this data immutable or unchanging.

To keep our arrays immutable, or unchanging, we are going to have to learn to use functions like map and filter. To keep objects immutable, we will need to learn how to duplicate and assign objects. You will need to be able to abstract objects from arrays with Object.keys and arrays from objects with array.reduce. All of this is pure JavaScript. All of this JavaScript is covered in this book. These practices can apply to any JavaScript application not just React.

It is our hope that you can use this book to skip confusion in the learning process by jumping straight into functional JavaScript development with React. We hope to get you up to speed with the React ecosystem faster by approaching it this way.

React Ecosystem

As we mentioned, React is a library. It's not a part of any overarching framework. There is an ecosystem of popular libraries and design patterns that we can use when building web applications with React. We get to choose our user interface stack out of this ecosystem of libraries to create our own stack.

A good rule of thumb when working with React is to use only what you need. For a lot of apps, you can use just React. React can manage views and even the data that is used in the views. That is powerful enough alone to build many types of applications. To that end, you don't have to use a ton of complicated tools.

You can build small apps with just React. As apps grow in scale, you may find the need to incorporate other solutions. If you start with a full stack app, it might be overkill.

Overly complex tooling has become a common complaint about React and JavaScript in general. Much of this is borrowed trouble. If you don't need the tool or you hate working with it, there's a way to not use it or use a tool you do like.

If you need extra tools, here is what you need them for. The purpose of a build tool is to take tasks that you commonly perform (SASS to CSS, code linting, testing, Babel transpiling, etc.) and automate them. Each of these links will give you additional information on how to set up a project with that tool.

- Browserify - Allows you to use require modules like you might do in Node.js. Browserify is often used with Grunt and Gulp.
- Webpack - The build system we'll use in this book. It has a steeper learning curve than Browserify, but it is adopted widely in the community. Also, once you get the hang of it, you will likely enjoy working with it more.

These build systems are often used in React projects, but they are widely used on all sorts of projects that have nothing to do with React. There are also a variety of tools that are intended to support React-specific projects.

React Router

The React Router provides a framework for handling routing in React applications. The router helps handle routing in single page applications. The website loads one page, and the router manages navigation from page to page.

React Motion

A framework for creating animations in React. It does so by interpolating the values used in CSS transforms for things like x and y values changing over time, opacity changes, etc.

React Addons

A package of opt-in features that can be used to enhance React applications. These utilities help improve performance like PureRenderMixin and Perf, manage animations with CSSTransitionGroup, and write test cases with TestUtils.

Enzyme

An increasingly popular unit testing framework for React created by Airbnb. Enzyme allows you to use whichever assertion library or test runner you'd like including Karma, Mocha, Jest, and more.

Redux

Redux is an implementation of Flux. Though commonly associated with React, Redux is library agnostic. You can use Redux with any UI: Angular, Ember, jQuery, or regular JavaScript. It includes React Redux which contains the official React bindings for Redux. It also includes Redux Thunk, middleware that provides a way of handling more complex architectures in Redux applications.

React Fire (Firebase for React)

Firebase is a pre-built back-end for features like user authentication, data storage, and hosting. React Fire is the React-specific implementation of Firebase that can be integrated into React applications.

As mentioned at the beginning of this section, it can be helpful to have some awareness of the large ecosystem of React-related tools, but it's important not to get too bogged down in whatever the trendy new thing is. New tools are popping up and changing all the time. Use only what you need, and keep it simple.

Keeping up with the Changes

As we've mentioned, React is still new. It has reached a place where core functionality is fairly stable but even that can change. The tools, libraries, and ecosystem are still being discovered, and the development of these tools is also changing.

As changes are made to React and related tools, sometimes these are breaking changes. In fact, some of the future versions of these tools may break some of the example code in this book. You can still follow along with the code samples. We'll provide exact version information in the package.json file, so that you can install these packages at the correct version.

Beyond this book, you can stay on top of changes by following along with the official React blog. When new versions of React are released, the core team will write a detailed blog and changelog about what is new.

There are also a variety of popular React conferences that you can attend for the latest React information. If you can't attend these in-person, React conferences often release the talks on YouTube following the events.

- React Conf - Facebook sponsored conference in the Bay Area
- React Rally - Community conference in Salt Lake City
- ReactiveConf - Community conference in Bratislava, Slovakia
- React.Amsterdam - Community conference in Amsterdam

Working with the Files

In this section, we will discuss how to work with the files for this book and how to install some useful React tools.

File Repository

The GitHub repository associated with this book (https://github.com/moonhighway/learning-react) provides all of the code files organized by chapter. The repository is a mix of code files and JSBin samples. If you've never used JSBin before, it's an online code editor similar to CodePen and JSFiddle.

One of the main benefits of JSBin is that you can click the link and immediately start tinkering with the file. When you create or start editing a JSBin, it will generate a unique URL for your code sample.

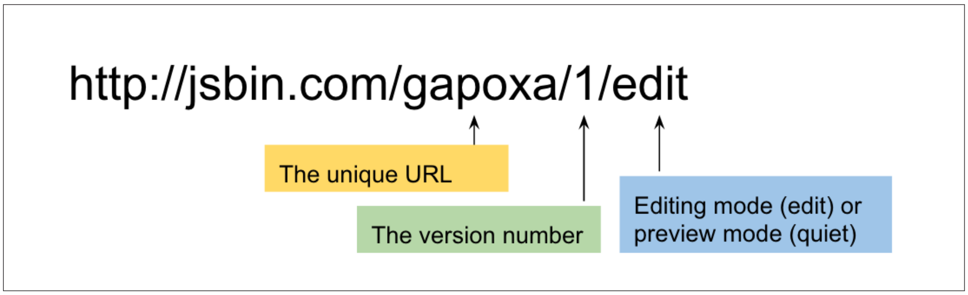

http://jsbin.com/gapoxa/1/edit
The unique URL
The version number
Editing mode (edit) or preview mode (quiet)

Figure 1-2. JSBin URL

The letters that follow jsbin.com represent the unique URL key. After the next slash is the version number. In the last part of the URL, there will be one of two words: edit for editing mode and quiet for preview mode.

React Developer Tools

There are several developer tools that can be installed as browser extensions or add-ons that you may find useful as well.

react-detector

The react-detector is a Chrome extension that lets you know which websites are using React and which are not. You can find the react-detector here: https://chrome.google.com/webstore/detail/react-detector/jaaklebbenondhkanegppcca-nebkdjlh?hl=en-US

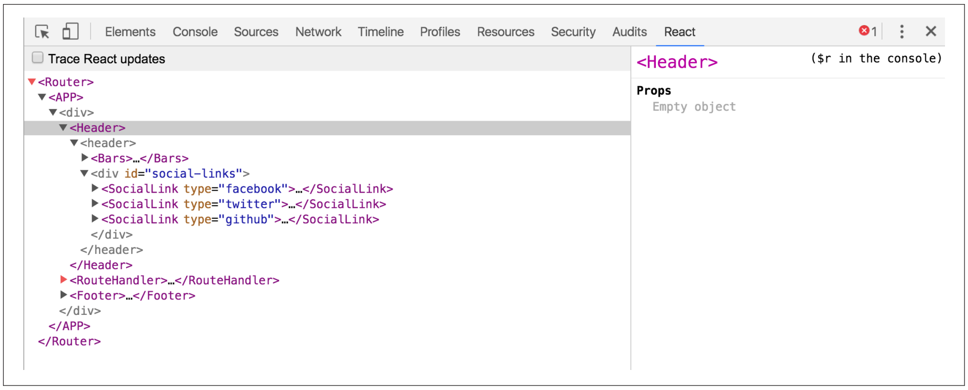
show-me-the-react

This is another tool that detects React as you browse the internet. It is available for both Firefox (https://github.com/insin/show-me-the-react) and Chrome (https://github.com/cymen/show-me-the-react).

React Developer Tools

The React Developer Tools are a plugin that can extend the functionality of the browser's developer tools. The React Developer Tools is installed as a new tab to let you view React elements.

If you prefer Chrome, you'll install as an extension:https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en. You can also install as an add-on for Firefox: https://addons.mozilla.org/en-US/firefox/addon/react-devtools/.



```
▼<Router>
 ▼<APP>
  ▼<div>
   ▼<Header>
    ▼<header>
     ▶<Bars>…</Bars>
     ▼<div id="social-links">
      ▶<SocialLink type="facebook">…</SocialLink>
      ▶<SocialLink type="twitter">…</SocialLink>
      ▶<SocialLink type="github">…</SocialLink>
      </div>
     </header>
    </Header>
   ▶<RouteHandler>…</RouteHandler>
   ▶<Footer>…</Footer>
   </div>
  </APP>
 </Router>
```

```
<Header>                ($r in the console)
Props
  Empty object
```

Figure 1-3. Viewing the React Developer Tools

Any time you see react-detector or show-me-the-react as active, you can open the developer tools and get an understanding of how React is being used on the site.

Installing Node.js

Node.js is JavaScript without the browser. It is a runtime environment used to build full-stack JavaScript applications. Node is open-source and can be installed on Windows, Mac, Linux, and other platforms.

You do not need to use Node to use React. We will be using Node in Chapter 12 when we build an Express server. However, when working with React, you need to use the Node package manager, nom, to install dependencies. This is automatically installed with the Node installation.

If you're not sure if Node.js is installed on your machine, you can open a Terminal or Command prompt window and type:

```
node -v

Output: v6.1.0
```

Ideally, you will have a Node version number of 4 or higher. If you type the command and see an error message that says "Command not found", Node.js is not installed. This can be done directly from the website at nodejs.org. Just go through the automated steps of the installer, and when you type in the 'node -v' command again, you'll see the version number.

Not only is React new and changing, but JavaScript is currently undergoing huge changes. We'll start by getting up to speed with the latest changes in JavaScript before diving into React.